Computing IV Sec 202: Project Portfolio

Jake Shick

Fall 2023

Contents

Time to Complete Portfolio: 10 hours

1 PS0: Hello SFML

1.1 Assignment Overview

Project 0 was the first assignment we were introduced to the SFML API in the semester. In this project, we were first tasked with re-creating the SFML tutorial that has us drawing a green circle in the center of the screen. After we completed this step, we then were tasked with completing assignment requirments which included drawing our own sprite to the window, making the sprite move, having the sprite move based on keystrokes, and adding one more feature. This assignment took around two hours to complete, as it was first time being introduced to any API and was a great first project to help guide me through the process.

1.2 Design Process

There were a few ways that I chose to implement this project that not only ensured that I was meeting the neccesary requirments of the project, but also would ensure a much easier time on future assignments that I would develop using the skills I learned in this one. I first found what image I wanted to use for my project. I decided to keep it orginal and found a png of a sprite can and decided that was what I would use. After a drew the image to the screen, I focused on having the sprite move and having it respond to keystokes. I decided to combine steps two and three by having the sprite freely move based on keystrokes in constant time. The last feature I added was having a backround song play while the window was open which I chose to be the hyrule fields theme from the Legend of Zelda Ocarina of Time. Afterwards, the finished project looked like the below figure which included the orginal circle from the tutorial of SFML, and a fully moveable sprite can that would only move left, right, up, and down based on keystrokes.

1.3 Key Algorithms, Data Structures, Objected-Oriented Designs, etc.

For this assignment, there weren't many elaborate data structures or algorithms used since the task was relatively basic, and a significant portion of the code was already provided. However, I incorporated object-oriented (OO) designs, particularly in creating my sprite and implementing its movement. Initially, I utilized a circle shape object to establish the initial tutorial step. Additionally, I employed a texture object to represent my customized sprite for the project. To manage the sprite's position during runtime, I employed two variables, x and y, both initialized to zero. Four boolean variables, initialized to false, served as flags to indicate when certain keystrokes were pressed and released. Subsequently, I created a sprite object to encapsulate the sprite can's texture and a music object to represent the Hyrule Fields theme. After verifying that the files opened correctly, I proceeded to draw the window, implemented if statements to detect key presses and releases, and updated the x and y values within my while loop to obtain the new position.

1.4 Screenshots

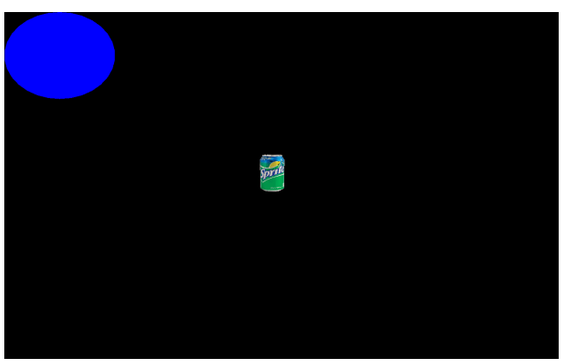

Figure 1: Window produced from running the tutorial code

1.5 What I Learned

This assignment enhanced my overall comprehension of how the SFML API functions work and familiarized me with its key features. Initially, I faced confusion regarding window space. However, completing this assignment proved invaluable in clarifying the behavior of window space and how variables and sprites are consistently affected in real-time as long as the window remains open. Despite its simplicity, I consider this assignment an excellent introductory project for gaining familiarity with SFML libraries.

1.6 Sources used in Projects

- SFML Tutorial Page: `https://www.sfml-dev.org/tutorials/2.4/start-linux.php`
- SFML Window Tutorial: `https://www.sfml-dev.org/tutorials/2.4/window-window.php`
- Lulu's Blog Page: `https://lucidar.me/en/sfml/sfml-part-4-moving-a-sprite-with-the-keyb`
- Sonar Systems Youtube: `https://www.youtube.com/watch?v=x8KGkaKQ6v8`

1.7 Codebase

```
CC = g++
CFLAGS = --std=c++17 -Wall -Werror -pedantic -g
LIB = -lsfml-graphics -lsfml-audio -lsfml-window -lsfml-system -
   lboost_unit_test_framework
# Your .hpp files
DEPS =
# Your compiled .o files
OBJECTS = main.o
# The name of your program
PROGRAM = sfml-app

.PHONY: all clean lint

all: $(PROGRAM)

%.o: %.cpp $(DEPS)
    $(CC) $(CFLAGS) -c $<

$(PROGRAM): main.o $(OBJECTS)
    $(CC) $(CFLAGS) -o $@ $^ $(LIB)

clean:
    rm *.o $(PROGRAM)

lint:
    cpplint *.cpp *.hpp
```

```
/*"Copyright [2023] <Jake Shick>"  [legal/copyright]*/
#include <iostream>
#include <SFML/Graphics.hpp>
#include <SFML/Audio.hpp>

int main() {
    int x = 0;
    int y = 0;

    bool up = false;
    bool down = false;
```

```
12      bool right = true;
13      bool left = false;
14
15      sf::CircleShape shape(100.f);
16      shape.setFillColor(sf::Color::Blue);
17
18      sf::Texture sprite_can;
19      if (!sprite_can.loadFromFile("sprite.png")) {
20          std::cout << "Did not open sprite correctly!" << std::endl;
21          return EXIT_FAILURE;
22      }
23
24      sf::Sprite a_literal_sprite_can;
25      a_literal_sprite_can.setTexture(sprite_can);
26      a_literal_sprite_can.setScale(0.1, 0.1);
27
28      sf::Music hyrule_field;
29      if (!hyrule_field.openFromFile("19 Hyrule Field Main Theme.flac")) {
30          std::cout << "Music File could not be opened!" << std::endl;
31          return EXIT_FAILURE;
32      }
33      hyrule_field.play();
34
35      sf::RenderWindow window(sf::VideoMode(1000, 800), "SFML works!");
36
37      while (window.isOpen()) {
38          sf::Event event;
39          while (window.pollEvent(event)) {
40              if (event.type == sf::Event::Closed || event.key.code == sf::
    Keyboard::Escape)
41                  window.close();
42              else if (event.type == sf::Event::KeyPressed) {
43                  if (event.key.code == sf::Keyboard::Up) {
44                      up = true;
45                  }
46                  else if (event.key.code == sf::Keyboard::Down) {
47                      down = true;
48                  }
49                  else if (event.key.code == sf::Keyboard::Left) {
50                      left = true;
51                  }
52                  else if (event.key.code == sf::Keyboard::Right) {
53                      right = true;
54                  }
55              }
56
57              if (event.type == sf::Event::KeyReleased) {
58                  if (event.key.code == sf::Keyboard::Up)
59                      up = false;
60                  else if (event.key.code == sf::Keyboard::Down)
61                      down = false;
62                  else if (event.key.code == sf::Keyboard::Left)
63                      left = false;
64                  else if (event.key.code == sf::Keyboard::Right)
65                      right = false;
66              }
67          }
68
69
```

```
70          window.clear();
71          if (left)
72              x--;
73
74          if (right)
75              x++;
76
77          if (up)
78              y--;
79
80          if (down)
81              y++;
82
83          a_literal_sprite_can.setPosition(x, y);
84          window.draw(a_literal_sprite_can);
85          window.draw(shape);
86          window.display();
87      }
88      return 0;
89  }
```

2 PS1:Linear Feedack Shift Register and Photo Magic

2.1 Assignment Overview

Project 1 had us create a Linear Feedback Shift Register that would essentially produce a random number based on a 16 bit string which we could then use in order to encrypt and decrypt images. We would get this number by XORing the 15th, 13th, 12th, and 10th positions of our binary string, remove the number at the left most place by pushing all numbers down by one to the left, and appending the new number we got from the first step to the begining of the binary string. After we got the number, we could then alter an image by extracting the pixels from the image and altering the pixel number.

2.2 Design Process

This assignment was presented in two parts. Part A focused on building my linear feedback register and having it output the correct binary string and correct number that resulted in calling either the step or generate function. The step function takes no paramters and would XOR the above specified numbers and output either a zero or one based on whether or the final XOR resulted in a zero or one. The generate function on the other hand calls step a n number of times and outputs the binary string along with the the number that results in reading n characters in the binary string. Part B used our linear feedback register we built in part A to decrypt an image of a cat into a series of multicolored pixels that were obtained by XORing the integer value of the pixels by the number generated by my generate function from part A. This was all done inside a new function called transform which did exactly that, transform the image into a series of pixels. We could then pass in the new pixelated image back into our trasnform function to get the orginal image back. At the end I creaded a test file which used functions from the boost library in order to test some of my functions.

2.3 Key Algorithms, Data Structures, Objected-Oriented Designs, etc.

I created a class FibLFSR, to represent our string of a binary string and count of generate, along with our methods that XORed the correct numbers in the binary string, shift the numbers of the string when either step or generate was called, and get the number that resulted in either generate or step being called. I didn't really use any data structures besides my objects I created in my main function to set the image, but I used an algorithm in my get seed function that would get the decimal number by looking at each individual bit in the string and checking to see if it was a one or zero. If it was a one, the number was added and the base was then changed to reflect what bit it was pointing to next. After exiting the for loop, the decimal number was returned to the caller.

2.4 Screenshots

Figure 2: Before Transform Called

Figure 3: After Transform Called

2.5 What I Learned

Before doing this assignment, I struggled with the concept of XOR and bitwise shifting, but after completing this project, I feel much more confident in my abilities to work with not only bitwise operators, but how XOR functionality works in C++. Along with this, I also learned much about data encyption and generating random numbers, a concept that I only knew a little about from past assignments from other computer science courses.

2.6 Sources used in Projects

- SFML Resize Image: `https://en.sfml-dev.org/forums/index.php?topic=24954.0`
- SFML Setting Pixels: `https://www.sfml-dev.org/documentation/2.6.0/classsf_1_1Image.php`
- Bitwise Shifting: `https://www.geeksforgeeks.org/bitwise-operators-in-c-cpp/`
- BOOST Library Tests: `https://stackoverflow.com/questions/26652904/boost-check-equal-a`

2.7 Codebase

```
CC = g++
CFLAGS = --std=c++17 -Wall -Werror -pedantic -g
LIB = -lsfml-graphics -lsfml-audio -lsfml-window -lsfml-system -
   lboost_unit_test_framework
# Your .hpp files
DEPS = FibLFSR.hpp PhotoMagic.hpp
# Your compiled .o files
OBJECTS = FibLFSR.o  PhotoMagic.o
# The name of your program
PROGRAM = PhotoMagic

.PHONY: all clean lint

all: $(PROGRAM) test $(PROGRAM).a

%.o: %.cpp $(DEPS)
    $(CC) $(CFLAGS) -c $<

$(PROGRAM): main.o $(OBJECTS)
    $(CC) $(CFLAGS) -o $@ $^ $(LIB)

test:  test.o $(OBJECTS)
    $(CC) $(CFLAGS) -o $@ $^ $(LIB)

$(PROGRAM).a: $(OBJECTS)
    ar rcs $@ $^

clean:
    rm *.o $(PROGRAM) test $(PROGRAM).a

lint:
    cpplint *.cpp *.hpp
```

```
// Copyright 2023
// By Jake Shick
// main.cpp for PS1

#include <iostream>
#include <string>
```

```cpp
#include "FibLFSR.hpp"
#include "PhotoMagic.hpp"
#include <SFML/System.hpp>
#include <SFML/Window.hpp>
#include <SFML/Graphics.hpp>

int main(int argc, char** argv) {
    sf::Image image;
    if (!image.loadFromFile(argv[1])) {
        return -1;
    }
    FibLFSR lfsr(argv[3]);

    sf::Vector2u size1 = image.getSize();
    sf::RenderWindow window1(sf::VideoMode(size1.x, size1.y), "Orginal");
    sf::RenderWindow window2(sf::VideoMode(size1.x, size1.y), "Decrypt");

    sf::Texture texture;
    texture.loadFromImage(image);

    sf::Sprite sprite;
    sprite.setTexture(texture);

    PhotoMagic::transform(image, &lfsr);

    sf::Texture texture2;
    texture2.loadFromImage(image);

    sf::Sprite sprite2;
    sprite2.setTexture(texture2);

    while (window1.isOpen() && window2.isOpen()) {
        sf::Event event;
        while (window1.pollEvent(event)) {
            if (event.type == sf::Event::Closed)
            window1.close();
        }
        while (window2.pollEvent(event)) {
            if (event.type == sf::Event::Closed)
                window2.close();
        }
        window1.clear();
        window1.draw(sprite);
        window1.display();
        window2.clear();
        window2.draw(sprite2);
        window2.display();
}


    // fredm: saving a PNG segfaults for me, though it does properly
    //   write the file
    if (!image.saveToFile(argv[2]))
        return -1;
    return 0;
}
```

```cpp
// Copyright 2023
// By Jake Shick
// FibLFSR.cpp for PS1
```

```
4
5  #include "FibLFSR.hpp"
6  #include <iostream>
7  #include <string>
8  #include <sstream>
9
10 const int zero = 0;
11 const int two = 2;
12 const int three = 3;
13 const int five = 5;
14 const int fifteen = 15;
15
16 FibLFSR::FibLFSR():FibLFSR("1011011000110110") {}
17
18 FibLFSR::FibLFSR(std::string seed) {
19     bits = seed;
20 }
21
22 int FibLFSR::step()  {
23     int temp = 1;
24     temp = XOR();
25     set_bits(temp);
26     int num = get_seed(count);
27     return num;
28 }
29
30 int FibLFSR::generate(int n) {
31     count = n;
32     int num = 0;
33     for (int i = 0; i < n; i++) {
34         num = step();
35     }
36     return num;
37 }
38
39 void FibLFSR::set_bits(int temp) {
40     for (int i = 0; i < fifteen; i++) {
41         bits[i] = bits[i + 1];
42     }
43
44     bits[fifteen] = temp + '0';
45 }
46
47 int FibLFSR::XOR() {
48     int temp = bits[zero] ^ bits[two];
49     temp = temp ^ bits[three];
50     temp = temp ^ bits[five];
51     return temp;
52 }
53 int FibLFSR::get_seed(int n) {
54     std::string test;
55     int base = 1, decimal = 0;
56
57     test = bits.substr(bits.length() - n);
58     for (int i = test.length() - 1; i >= 0; i--) {
59         if (test[i] == '1')
60             decimal += base;
61         base = base * 2;
62     }
```

```
    return decimal;
}
std::ostream& operator<<(std::ostream& out, const FibLFSR& lfsr) {
    std::string temp = lfsr.bits;
    out << temp;
    return out;
}
```

```
// Copyright 2023
// By Jake Shick
// FibLFSR.hpp for PS1

#pragma once
#include <string>

class FibLFSR {
 public:
    FibLFSR();
// Constructor to create LFSR with the given initial seed
    explicit FibLFSR(std::string seed);
// Simulate one step and return the new bit as 0 or 1
    int step();
// Simulate k steps and return a k-bit integer
    int generate(int k);

    std::string get_bits();

    int get_count();

    void set_bits(int temp);

    int XOR();

    int get_seed(int k);

    friend std::ostream& operator<<(std::ostream&, const FibLFSR& lfsr);

 private:
    std::string bits;
    int count = 1;
};
```

```
// Copyright 2023
// By Jake Shick
// FibLFSR.cpp for PS1

#include <iostream>
#include "FibLFSR.hpp"
#include "PhotoMagic.hpp"

// const int color_chanel_size;
namespace PhotoMagic {
    void transform(sf::Image& image, FibLFSR* lfsr) {
        // p is a pixelimage.getPixel(x, y);
        sf::Color p;
        const sf::Vector2 orgianl_size = image.getSize();
        int horizontal = orgianl_size.x;
        int vertical = orgianl_size.y;
        // create photographic negative image of upper-left 200 px square
        for (int x = 0; x < horizontal; x++) {
```

```
            for (int y = 0; y < vertical; y++) {
                p = image.getPixel(x, y);
                p.r = 255 - p.r;
                int8_t red = p.r ^ lfsr->generate(8);
                p.g = 255 - p.g;
                int8_t green = p.g ^ lfsr->generate(8);
                p.b = 255 - p.b;
                int8_t blue = p.b ^ lfsr->generate(8);
                sf::Color new_color(red, green, blue);
                image.setPixel(x, y, new_color);
            }
        }
    }
}  // namespace PhotoMagic
```

```
// Copyright 2023
// By Jake Shick
// FibLFSR.cpp for PS1

#include <iostream>
#include "FibLFSR.hpp"
#include <SFML/System.hpp>
#include <SFML/Window.hpp>
#include <SFML/Graphics.hpp>

namespace PhotoMagic {
// Transforms image using FibLFSR
    void transform(sf::Image& image, FibLFSR* lfsr);
// Display an encrypted copy of the picture, using the LFSR to do the
    encryption
}
```

```
// Copyright 2022
// By Dr. Rykalova
// Editted by Dr. Daly
// Also editted by Jake Shick
// test.cpp for PS1
// updated 5/12/2022

#include <iostream>
#include <string>
#include "FibLFSR.hpp"
#define BOOST_TEST_DYN_LINK
#define BOOST_TEST_MODULE Main
#include <boost/test/unit_test.hpp>

BOOST_AUTO_TEST_CASE(testStepInstr1) {
  FibLFSR l("1011011000110110");
  BOOST_REQUIRE_EQUAL(l.step(), 0);
  BOOST_REQUIRE_EQUAL(l.step(), 0);
  BOOST_REQUIRE_EQUAL(l.step(), 0);
  BOOST_REQUIRE_EQUAL(l.step(), 1);
  BOOST_REQUIRE_EQUAL(l.step(), 1);
  BOOST_REQUIRE_EQUAL(l.step(), 0);
  BOOST_REQUIRE_EQUAL(l.step(), 0);
  BOOST_REQUIRE_EQUAL(l.step(), 1);
}

BOOST_AUTO_TEST_CASE(testStepInstr2) {
  FibLFSR l2("1011011000110110");
```

```
29   BOOST_REQUIRE_EQUAL(l2.generate(9), 51);
30 }
31
32 // Makes sure generate does pass if value is in ascii table form
33 BOOST_AUTO_TEST_CASE(testStepInstr3) {
34   FibLFSR l3("1011011000110110");
35   BOOST_CHECK_EQUAL(l3.generate(9), '3');
36 }
37
38 // Makes sure step does not work if value is inncorrect
39 BOOST_AUTO_TEST_CASE(testStepInstr4) {
40   FibLFSR l4("1011011000110110");
41   BOOST_REQUIRE_EQUAL(!l4.step(), 1);
42 }
43
44 // Makes sure program does not work correctly if enetered garbage characters
45 BOOST_AUTO_TEST_CASE(testStepInstr5) {
46   FibLFSR l5("a0bc0d1000110110");
47   BOOST_REQUIRE_EQUAL(!l5.generate(4), 1);
48 }
49
50 // Makes sure program will return correct XOR value
51 BOOST_AUTO_TEST_CASE(testStepInstr6) {
52   FibLFSR l6("1011011000110110");
53   BOOST_REQUIRE_EQUAL(l6.XOR(), 0);
54 }
```

3 PS2: Recursive Graphics (Hexaflake)

3.1 Assignment Overview

Project 2 mimiced something we went over in class called the Sierpinski Triangle. Essensially how it worked was that you had a base function which would draw the base shape of Triangle, and then you would have a helper recursive function which would draw a sequence of specifcally ordered triangles either outside or sinside our orginal one. The size of these triangles would be altered depending on the depth of recursion to account for spillage when larger depths were entered. Our Hexaflake project was very similiar to this, but instead of having hexagons drawn inside our base, we had the base change depending on the depth and had the child hexagons drawn around our original.

3.2 Design Process

For this, I started out by drawing my hexagons on a sheet of paper and hand writing the math in order to figure out how to get the position of each child hexagon. By doing this, I understood the math a lot better when actually designing this program, which helped when figuring out problems that would arise later into its production. I first created the base hexagon to scale somewhat with the window so when I called my helper function, their would be no overlap or out of bounds errors. Once I did this, I focused on drawing the children hexagon by finding the new positions of where the next hexagon will be based on the cosine and sine of the hexagons six equilateral triangles inside the base. Once this is calculated, we can go around our base hexagon in order to find each new child hexagons position.

3.3 Key Algorithms, Data Structures, Objected-Oriented Designs, etc.

For data structures, I used a Vector2f to represent the window's center that would useful when looking for where the center of my window was, which was more useful for drawing the base hexagon then anything. I also created multiple Vector2f objects in order to represent the center of multiple child hexagons in order to calculate what their child hexagons would look like if the depth went beyond 1.

3.4 Screenshots

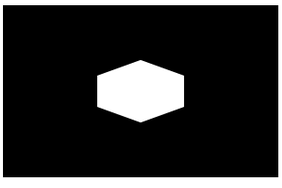

Figure 4: Iteration 0

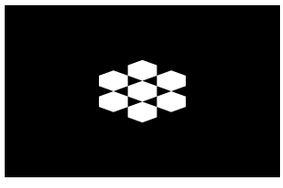

Figure 5: Iteration 1

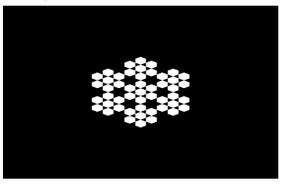

Figure 6: Iteration 2

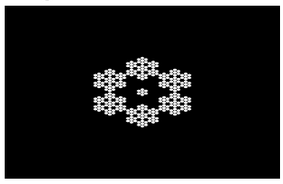

Figure 7: Iteration 3

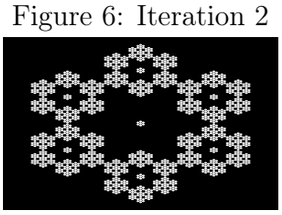

Figure 8: Iteration 4

3.5 What I Learned

This assignment was without a doubt the toughest for me to grasp. Recursion is always hard to visualize since the callers constantly shift back and forth, but this assignment helped really iron out any problems I may have previously had while trying to understand or fully comprehend recursion.

3.6 Sources used in Projects

- No Sources Used.

3.7 Challenges

For myself, this assignment posed the hardest in the entire semester. Orginally I did not do the math on paper before attempting to solve this and the outcome was less than optimal. I was constantly fighting with this one in order to get the most simple outcome, and in the end I got the code to work somewhat but not as well as I would have liked, with much of it being hardcoded. I plan on fully completling this one in the future in order to better grasp for future assignments that may rely on knowledge learned in this one.

3.8 Codebase

```
CC = g++
CFLAGS = --std=c++17 -Wall -Werror -pedantic -g
LIB = -lsfml-graphics -lsfml-audio -lsfml-window -lsfml-system -
   lboost_unit_test_framework
# Your .hpp files
DEPS = hexa.hpp
# Your compiled .o files
OBJECTS = hexa.o
# The name of your program
PROGRAM = Hexa

.PHONY: all clean lint

all: $(PROGRAM)

%.o: %.cpp $(DEPS)
    $(CC) $(CFLAGS) -c $<

$(PROGRAM): main.o $(OBJECTS)
    $(CC) $(CFLAGS) -o $@ $^ $(LIB)

Test: Test.o $(OBJECTS)
    $(CC) $(CFLAGS) -o $@ $^ $(LIB)

clean:
    rm *.o $(PROGRAM)

lint:
    cpplint *.cpp *.hpp
```

```
// Copyright 2023
// By Jake Shick
// main.cpp for PS2
#include <cmath>
#include <SFML/Graphics.hpp>
#include "hexa.hpp"

int main(int argc, char** argv) {
```

```
    unsigned int L = std::stoi(argv[1]);
    unsigned int N = std::stoi(argv[2]);
    unsigned int windowSize = (5.5 * L);
    sf::RenderWindow window(sf::VideoMode(windowSize, windowSize), "SFML
   works!");
    sf::Vector2f center(windowSize / 2, windowSize / 2);
    while (window.isOpen()) {
        sf::Event event;
        while (window.pollEvent(event)) {
            if (event.type == sf::Event::Closed)
                window.close();
        }

        window.clear(sf::Color::Black);
        drawSnowflake(window, N, center, L);
        window.display();
    }
    return 0;
}
```

```
// Copyright 2023
// By Jake Shick
// main.cpp for PS2
#include "hexa.hpp"
#include <cmath>

const double DEG_TO_RAD = M_PI / 6;

void drawSnowflake(sf::RenderTarget& window, int depth, sf::Vector2f center,
    float length) {
    float radius = length;
    if (depth >= 1)
        radius = length / 3;
    if (depth >= 2)
        radius = length / 8;
    if (depth >= 3)
        radius = length / depth / 8;
    sf::CircleShape Hexagon(radius, 6);
    Hexagon.setOrigin(radius, radius);
    sf::Vector2f pt = center;
    Hexagon.setPosition(pt);
    Hexagon.setFillColor(sf::Color::White);
    window.draw(Hexagon);
    drawHexagonHelper(window, depth, center, radius, 1);
}

void drawHexagonHelper(sf::RenderTarget& window, int depth,
sf::Vector2f center, float radius, double thetaDeg) {
    int count = 0;
    if (depth != 0) {
        sf::CircleShape triangle(radius, 6);
        triangle.setOrigin(radius, radius);
        sf::Vector2f pt = center;
        triangle.setPosition(pt);
        triangle.setFillColor(sf::Color::White);
        window.draw(triangle);
        for (int i = 1; i <= 11; i += 2) {
            sf::Vector2f pt = center;
            sf::CircleShape Hexagon(radius, 6);
            Hexagon.setOrigin(radius, radius);
```

```
            pt.x +=(2 * radius * cos(i * DEG_TO_RAD));
            pt.y +=(2 * radius * sin(i * DEG_TO_RAD));
            Hexagon.setPosition(pt);
            Hexagon.setFillColor(sf::Color::White);
            window.draw(Hexagon);
        }
        if (depth == 1) {
            count = 2;
        } else if (depth == 2) {
            count = 6;
        } else if (depth == 3) {
            count = 18;
        } else if (depth == 4) {
            count = 54;
        }
        sf::Vector2f center1 = center;
        sf::Vector2f center2 = center;
        sf::Vector2f center3 = center;
        sf::Vector2f center4 = center;
        sf::Vector2f center5 = center;
        sf::Vector2f center6 = center;
        center1.x +=  (count * radius * cos(thetaDeg * DEG_TO_RAD));
        center1.y +=  (count * radius * sin(thetaDeg * DEG_TO_RAD));
        center2.x +=  (count * radius * cos((thetaDeg + 2) * DEG_TO_RAD));
        center2.y +=  (count * radius * sin((thetaDeg + 2) * DEG_TO_RAD));
        center3.x +=  (count * radius * cos((thetaDeg + 4) * DEG_TO_RAD));
        center3.y +=  (count * radius * sin((thetaDeg + 4) * DEG_TO_RAD));
        center4.x +=  (count * radius * cos((thetaDeg + 6) * DEG_TO_RAD));
        center4.y +=  (count * radius * sin((thetaDeg + 6) * DEG_TO_RAD));
        center5.x +=  (count * radius * cos((thetaDeg + 8) * DEG_TO_RAD));
        center5.y +=  (count * radius * sin((thetaDeg + 8) * DEG_TO_RAD));
        center6.x +=  (count * radius * cos((thetaDeg + 10) * DEG_TO_RAD));
        center6.y +=  (count * radius * sin((thetaDeg + 10) * DEG_TO_RAD));
        drawHexagonHelper(window, depth - 1, center1, radius, thetaDeg);
        drawHexagonHelper(window, depth - 1, center2, radius, thetaDeg + 2);
        drawHexagonHelper(window, depth - 1, center3, radius, thetaDeg + 4);
        drawHexagonHelper(window, depth - 1, center4, radius, thetaDeg + 6);
        drawHexagonHelper(window, depth - 1, center5, radius, thetaDeg + 8);
        drawHexagonHelper(window, depth - 1, center6, radius, thetaDeg + 10)
    ;
     }
}
```

```
// Copyright 2023
// By Jake Shick
// main.cpp for PS2
#pragma once

#include <iostream>
#include <SFML/Graphics.hpp>

void drawSnowflake(sf::RenderTarget& window, int depth, sf::Vector2f center,
    float length);
void drawHexagonHelper(sf::RenderTarget& window,
int depth, sf::Vector2f center, float radius, double thetaDeg);
```

4 PS3: N-Body Simulation

4.1 Assignment Overview

For project 3, the assignment was to read in from a file called planets.txt. This file started off by listing the total number of planets, along with the radius of the entire universe. It then listed the six features of the planet, the x position, y position, x velocity, y velocity, mass, and the name of the planet's image file. We were then tasked with drawing the positions of the planets to scale with the window in sfml for part A and then have the planets rotate around the sun in part B.

4.2 Design Process

Like project 2, the first thing I did was figure out the math for this problem. The hardest parts about these assignments for me is figuring out how the math actually works, so it's ususally the first thing I try to figure out. Once I understood that by dividing the x position with radius of the universe and then multiplying that by half the size of the window, the rest of the assignment was the actual implementation. I decided to use two classes for this assignment as recommended by our professor, one to represent our universe(the number of planets and the radius) and one to represent the celestial bodies (the x and y positions, velocites, masses and name). This finished up part A, but for part B I had to include movement to the planets.

To do this, I decided to create two more functions that would handle the movement, my move accelerate and my move planets functions. For my move accelerate function, this handled finding the acceleration of each planet and updating the velocity accordingly. My move planets function simply added the velocity to the x and y position while multipling the velocity by delta t(the change in time) in order to get the new position.

4.3 Key Algorithms, Data Structures, Objected-Oriented Designs, etc.

For my data structues, I decided to use a vector of shared pointers to my celestial body class inside my universe class. This was because we wanted to avoid having to manually allocate the memory needed in case we wanted to open different files that contained more then five planets. By making them shared pointers, we were able to access the methods defined in the Celestial Body class in order to pass in the radius of our universe, along with the change in time. A key algorithm I used was to make new pointers in the istream operator for as long as their were still planets. By doing this, we didn't need to worry about de-allocating memory, and we knew during runtime how many pointers we would have.

4.4 Screenshots

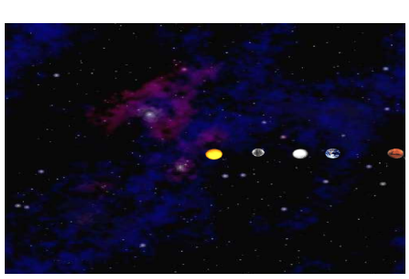

Figure 9: Before Transform Called

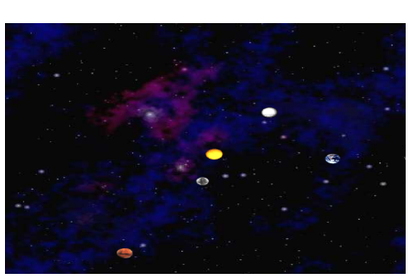

Figure 10: After Transform Called

4.5 What I Learned

This project taught me a lot about shared pointers since I had never really used them up to this point. Figuring out how they worked and more importantly how to call methods from them using the arrow operator was without a doubt my biggest struggle and achievement once I solved it.

4.6 Sources used in Projects

- How to Read into a file using istream: https://stackoverflow.com/questions/6255339/checking-if-a-file-opened-successfully-with-ifstream
- How to use shared pointers: https://www.geeksforgeeks.org/auto_ptr-unique_ptr-shared_ptr-weak_ptr-in-cpp/

4.7 Codebase

```
CC = g++
CFLAGS = --std=c++17 -Wall -Werror -pedantic -g
LIB = -lsfml-graphics -lsfml-audio -lsfml-window -lsfml-system -
   lboost_unit_test_framework
# Your .hpp files
DEPS = Universe.hpp CelestialBody.hpp
# Your compiled .o files
OBJECTS = Universe.o CelestialBody.o
# The name of your program
PROGRAM = NBody

.PHONY: all clean lint

all: $(PROGRAM) test $(PROGRAM).a

%.o: %.cpp $(DEPS)
    $(CC) $(CFLAGS) -c $<

%.o: %.cpp $(DEPS)
    $(CC) $(CFLAGS) -c $<

$(PROGRAM): main.o $(OBJECTS)
    $(CC) $(CFLAGS) -o $@ $^ $(LIB)

test:  test.o $(OBJECTS)
    $(CC) $(CFLAGS) -o $@ $^ $(LIB)

$(PROGRAM).a: $(OBJECTS)
    ar rcs $@ $^

clean:
    rm *.o $(PROGRAM) test $(PROGRAM).a

lint:
    cpplint *.cpp *.hpp
```

```
// Copyright 2023
// By Jake Shick
// main.cpp for PS3
#include <iostream>
#include <sstream>
#include <SFML/Graphics.hpp>
#include "Universe.hpp"
#include "CelestialBody.hpp"

int main(int argc, char** argv) {
    sf::RenderWindow window(sf::VideoMode(height, width), "Our Universe!");
    sf::Texture texture;
    double t = 0.0;
    double T = std::stod(argv[1]);
```

```
15      double delta_t = std::stod(argv[2]);
16
17      if (!texture.loadFromFile("starfield.jpg")) {
18          std::cout << "Could not open file!" << std::endl;
19      }
20      sf::Sprite sprite;
21      sprite.setTexture(texture);
22
23      Universe test;
24      std::cin >> test;
25      while (window.isOpen()) {
26          sf::Event event;
27          while (window.pollEvent(event)) {
28              if (event.type == sf::Event::Closed)
29                  window.close();
30          }
31          if (t < T) {
32              test.step(delta_t);
33              t += delta_t;
34          }
35          window.clear();
36          window.draw(sprite);
37          window.draw(test);
38          window.display();
39      }
40      std::cout << test;
41      return 0;
42  }
43
44  /*
45  CelestialBody earth;
46      CelestialBody mars;
47      CelestialBody mercury;
48      CelestialBody sun;
49      CelestialBody venus;
50
51      std::cin >> earth;
52      std::cout << earth;
53      earth.set_Planets(window.getSize().x, window.getSize().y, test.
        get_radius());
54
55      std::cin >> mars;
56      std::cout << mars;
57      mars.set_Planets(window.getSize().x, window.getSize().y, test.get_radius
        ());
58
59      std::cin >> mercury;
60      std::cout << mercury;
61      mercury.set_Planets(window.getSize().x, window.getSize().y, test.
        get_radius());
62
63      std::cin >> sun;
64      std::cout << sun;
65      sun.set_Planets(window.getSize().x, window.getSize().y, test.get_radius
        ());
66
67      std::cin >> venus;
68      std::cout << venus;
69      venus.set_Planets(window.getSize().x, window.getSize().y, test.
```

```
    get_radius());
*/
```

```
// Copyright 2023
// By Jake Shick
// CelestialBody.cpp for PS3
#include "CelestialBody.hpp"

std::istream &operator >>(std::istream &cin, CelestialBody &c1) {
    cin >> c1.xpos >> c1.ypos >> c1.xvel >> c1.yvel >> c1.mass >> c1.planet;
    if (!c1.texture_planet.loadFromFile(c1.planet)) {
        std::cout << "Failed to load \"c1.planet\"" << std::endl;
        return cin;
    }
    c1.sprite_planet.setTexture(c1.texture_planet);
    return cin;
}

std::ostream &operator <<(std::ostream &out, CelestialBody &c1) {
    out << c1.xpos << " ";
    out << c1.ypos << " ";
    out << c1.xvel << " ";
    out << c1.yvel << " ";
    out << c1.mass << " ";
    out << c1.planet << std::endl;
    return out;
}

double CelestialBody::get_xVelocity() {
    return xvel;
}
double CelestialBody::get_yVeloicty() {
    return yvel;
}

void CelestialBody::set_Planets(int window_x, int window_y, double radius) {
    double rxpos = ((xpos / radius) * (window_x / 2) + (window_x / 2));
    double rypos = (-(ypos / radius) * (window_y / 2) + (window_y / 2));
    sprite_planet.setPosition(rxpos, rypos);
}

std::string CelestialBody::get_planet() {
    return planet;
}

// sf::Sprite CelestialBody::get_sprite() {
//     return sprite_planet;
// }
double CelestialBody::get_xPos() {
    return xpos;
}
double CelestialBody::get_yPos() {
    return ypos;
}
void CelestialBody::draw(sf::RenderTarget& window, sf::RenderStates states)
    const {
    window.draw(sprite_planet, states);
}

void CelestialBody::move_accelerate(CelestialBody const &p, double t) {
```

```
    double dx = p.xpos - xpos;
    double dy = p.ypos - ypos;

    double r2 = (dx * dx) + (dy * dy);
    double r = sqrt(r2);
    double F = ((G * mass * p.mass) / r2);

    double force_x = F * dx / r;
    double force_y = F * dy / r;

    double accelX = force_x / mass;
    double accelY = force_y / mass;

    xvel += (accelX * t);
    yvel += (accelY * t);
}

void CelestialBody::move_planets(double t) {
    xpos += (xvel * t);
    ypos += (yvel * t);
}
```

```
// Copyright 2023
// By Jake Shick
// CelestialBody.hpp for PS3
#pragma once
#include <iostream>
#include <string>
#include <cmath>
#include <SFML/Graphics.hpp>
#include <SFML/Audio.hpp>


const double G = 6.67e-11;
const int Height = 500;
const int Width = 500;
const double radius = 2.50e+11;

class CelestialBody : public sf::Drawable {
 public:
  void set_Planets(int window_x, int window_y, double radius);
  void move_accelerate(CelestialBody const &p, double t);
  void move_planets(double t);

  double get_xPos();
  double get_yPos();
  double get_xVelocity();
  double get_yVeloicty();
  double get_mass();
  std::string get_planet();
  // sf::Sprite get_sprite();

  friend std::istream &operator >>(std::istream &cin, CelestialBody &c1);
  friend std::ostream &operator <<(std::ostream &out, CelestialBody &c1);

 private:
  void virtual draw(sf::RenderTarget& window, sf::RenderStates states) const
    ;
  sf::Texture texture_planet;
  sf::Sprite sprite_planet;
```

```
  double xpos;
  double ypos;
  double xvel;
  double yvel;
  double mass;
  std::string planet;
};
```

```
// Copyright 2023
// By Jake Shick
// Universe.cpp for PS3
#include "Universe.hpp"

double Universe::get_radius() {
    return radius;
}
int Universe::get_num_planets() {
    return num_planets;
}

void Universe::step(double t) {
    for (auto c : planets) {
        for (auto p : planets)  {
            if (c != p) {
                c->move_accelerate(*p, t);
            }
        }
    }
    for (auto c : planets) {
        c->move_planets(t);
        c->set_Planets(Width, Height, get_radius());
    }
}
std::istream &operator >>(std::istream &cin, Universe& p1) {
    cin >> p1.num_planets;
    cin >> p1.radius;
    for (int i = 0; i < p1.num_planets; i++) {
        std::shared_ptr<CelestialBody> temp = std::make_shared<CelestialBody
   >();
        cin >> *temp;
        p1.planets.push_back(temp);
    }
    return cin;
}

std::ostream &operator <<(std::ostream &out, Universe& p1) {
    out << p1.num_planets << std::endl;
    out << std::scientific;
    out << p1.radius << std::endl;
    for (auto &v : p1.planets) {
        out << *v;
    }
    return out;
}

void Universe::draw(sf::RenderTarget& window, sf::RenderStates states) const
    {
    for (auto &v : planets) {
        window.draw(*v, states);
    }
```

```
}
```

```
// Copyright 2023
// By Jake Shick
// Universe.hpp for PS3
#pragma once
#include <iostream>
#include <vector>
#include <sstream>
#include <fstream>
#include <string>
#include <memory>
#include "CelestialBody.hpp"

const int height = 500;
const int width = 500;

class Universe : public sf::Drawable {
 public:
double get_radius();
int get_num_planets();
void step(double t);
friend std::istream &operator >>(std::istream &cin, Universe &p1);
friend std::ostream &operator <<(std::ostream &out, Universe &p1);

 private:
void virtual draw(sf::RenderTarget& window, sf::RenderStates states) const;
int num_planets;
std::vector <std::shared_ptr<CelestialBody>> planets;
double radius;
};
```

```
// Copyright 2022
// By Dr. Rykalova
// Editted by Dr. Daly
// test.cpp for PS3
// updated 5/12/2022

#include <fstream>
#include <iostream>
#include <string>
#include <cmath>
#include "Universe.hpp"
#include "CelestialBody.hpp"
#define BOOST_TEST_DYN_LINK
#define BOOST_TEST_MODULE Main
#include <boost/test/unit_test.hpp>

BOOST_AUTO_TEST_CASE(test01) {
    Universe t;
    std::ifstream file;
    file.open("binary.txt");
    file >> t;
    BOOST_REQUIRE_EQUAL(t.get_radius(), 5.0e10);
    file.close();
}

BOOST_AUTO_TEST_CASE(test02) {
    Universe t;
    std::ifstream file;
```

```
    file.open("binary.txt");
    file >> t;
    BOOST_REQUIRE_EQUAL(t.get_num_planets(), 2);
    file.close();
}

BOOST_AUTO_TEST_CASE(test03) {
    Universe t;
    std::ifstream file;
    file.open("pluto.txt");
    file >> t;
    BOOST_REQUIRE_EQUAL(t.get_num_planets(), 13);
    file.close();
}

BOOST_AUTO_TEST_CASE(test04) {
    Universe t;
    std::ifstream file;
    file.open("pluto.txt");
    file >> t;
    BOOST_REQUIRE_EQUAL(t.get_radius(), 7.50e12);
    file.close();
}

BOOST_AUTO_TEST_CASE(test05) {
    Universe t;
    double time = 0.0;
    double T = 25000.0;
    double d_t = 25000.0;
    std::ifstream file;
    file.open("pluto.txt");
    file >> t;
    if (time < T) {
        t.step(d_t);
        time += d_t;
    }
    BOOST_CHECK_CLOSE(time, d_t, 0.1);
}

BOOST_AUTO_TEST_CASE(test06) {
    Universe t;
    double time = 0.0;
    double T = 50001.0;
    double d_t = 25000.0;
    std::ifstream file;
    file.open("pluto.txt");
    file >> t;
    if (time < T) {
        t.step(d_t);
        time += d_t;
    }
    BOOST_CHECK_CLOSE(time, d_t, 0.1);
}
```

5 PS4: Sokoban

5.1 Assignment Overview

With out a doubt, project 4 was my favorite of the semester. It took everything we had learned during the semester about SFML and had us actually create a fully functional game called Sokoban. The game has a player move around a pre-generated map that is built using a symbol table to represent the different textures/sprites. The player can move foward, left, right, or down depending on a number of factors like if there is no wall or no two boxes hitting, etc. Once the player moves all the boxes to their respective spots, the game ends with a theme and haults the player from moving at all unless they press R to reset the game to its orginal state.

5.2 Design Process

For part A, we were tasked with designing the basic UI for drawing the pre-generated map along with the player in all the respective positions. My first design choice was deciding how I would change the symbol table from an assortment of characters into a fully drawn gamespace. So the first thing I decided to do was load in my textures into private member variables in my newly created Sokoban class. This allowed me to access these textures anywhere in my program which would be useful when I needed to update the textures when my player was actually moving during part B. Once I handled that, I moved on to handle the way my board and player were drawn, which I'll talk more about below.

For part B, I ran into the problem of the player not knowing whether or not their was an object they could either go through or not not go through. I figured out that for everthing besides the ground, the player needed to know where everything was and for objects that could move, where they could possibly go. Once I understood this, I moved onto creating data structures in order to determine these factors.

5.3 Key Algorithms, Data Structures, Objected-Oriented Designs, etc.

The use of data structures for this project was absolutly essential for my design. I ended up using three vectors of Vector2i's to represent the wall position, filled ground , and crate position, along with another Vector2i to represent the players position. In the end, I created functions to check to see where the next position of either the player or boxes would be and move them there as long as there was space. I then used a previous position function to check and see if the player did move to somewhere they could not, and overide their decision to move there in the first place.

5.4 Screenshots

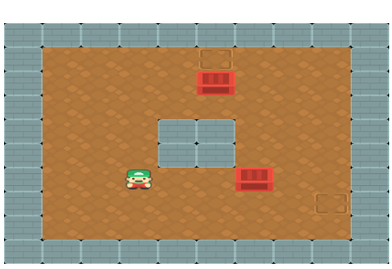

Figure 11: Intial UI

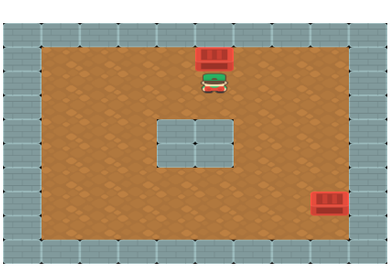

Figure 12: After player movement added

5.5 What I Learned

This project was one of my favorites while also being one of the most challenging, I stuggled a lot with figuring out how to check to see if something was blocking the player or the box. Once I solved this however, I was rewarded with a functional game that's not only pretty fun, but also very rewarding when won. Overall, I learned a lot more about the SFML built in vectors and all the best ways to traverse and use them.

5.6 Sources used in Projects

- Setting Texture Help: `https://www.sfml-dev.org/tutorials/2.6/graphics-sprite.php`
- Transparent Textures Help: `https://gamedev.stackexchange.com/questions/114405/problem-with-transparent-textures-in-sfml`

5.7 Codebase

```
CC = g++
CFLAGS = --std=c++17 -Wall -Werror -pedantic -g
LIB = -lsfml-graphics -lsfml-audio -lsfml-window -lsfml-system -
   lboost_unit_test_framework
# Your .hpp files
DEPS = Sokoban.hpp
# Your compiled .o files
OBJECTS = main.o Sokoban.o
# The name of your program
PROGRAM = Sokoban

.PHONY: all clean lint

all: $(PROGRAM) $(PROGRAM).a

%.o: %.cpp $(DEPS)
    $(CC) $(CFLAGS) -c $<

%.o: %.cpp $(DEPS)
    $(CC) $(CFLAGS) -c $<

$(PROGRAM): main.o $(OBJECTS)
    $(CC) $(CFLAGS) -o $@ $^ $(LIB)

$(PROGRAM).a: $(OBJECTS)
    ar rcs $@ $^

clean:
    rm *.o $(PROGRAM) $(PROGRAM).a

lint:
    cpplint *.cpp *.hpp
```

```
// Copyright 2023
// By Jake Shick
// main.cpp for PS4b
#include <iostream>
#include <fstream>
#include <SFML/Graphics.hpp>
#include <SFML/Audio.hpp>
#include "Sokoban.hpp"

int main(int argc, char* argv[]) {
    std::ifstream file;
    file.open(argv[1]);
    Sokoban t;
    file >> t;
    int h = t.height();
    int w = t.width();
    sf::Music happy_wheels;
```

```
    if (!happy_wheels.openFromFile("victory_sJDDywi.wav")) {
        std::cout << "Music File could not be opened!" << std::endl;
        return EXIT_FAILURE;
    }
    sf::RenderWindow window(sf::VideoMode(w * GRID_SIZE, h * GRID_SIZE), "Sokoban!");
    while (window.isOpen()) {
        sf::Event event;
        while (window.pollEvent(event)) {
            Direction player;
            if (event.type == sf::Event::Closed || event.key.code == sf::Keyboard::Escape) {
                window.close();
            }
            if (event.type == sf::Event::KeyPressed) {
                if (event.key.code == sf::Keyboard::Up || event.key.code == sf::Keyboard::W) {
                    player = Up;
                    t.movePlayer(player);
                }

                if (event.key.code == sf::Keyboard::Down || event.key.code == sf::Keyboard::S) {
                    player = Down;
                    t.movePlayer(player);
                }

                if (event.key.code == sf::Keyboard::Left || event.key.code == sf::Keyboard::A) {
                    player = Left;
                    t.movePlayer(player);
                }

                if (event.key.code == sf::Keyboard::Right || event.key.code == sf::Keyboard::D) {
                    player = Right;
                    t.movePlayer(player);
                }

                if (event.key.code == sf::Keyboard::R) {
                    t.setOriginal();
                }
                if (t.isWon()) {
                    happy_wheels.play();
                    std::cout << "CONGRATS!" << std::endl;
                }
            }
        }
        window.clear();
        window.draw(t);
        window.display();
    }
    return 0;
}
```

```
// Copyright 2023
// By Jake Shick
// main.cpp for PS4b
#include "Sokoban.hpp"
Sokoban::Sokoban() {
```

```
6       size_y = 0;
7       size_x = 0;
8       texture_ground.loadFromFile("ground_01.png");
9       texture_player.loadFromFile("player_05.png");
10      texture_wall.loadFromFile("block_06.png");
11      texture_crate.loadFromFile("crate_03.png");
12      texture_push_block.loadFromFile("ground_04.png");
13      texture_filled.loadFromFile("ground_04.png");
14  }
15
16  std::istream &operator >>(std::istream &cin, Sokoban &s1) {
17      cin >> s1.size_y >> s1.size_x;
18      s1.texture_ground.loadFromFile("ground_01.png");
19      s1.texture_player.loadFromFile("player_05.png");
20      s1.texture_wall.loadFromFile("block_06.png");
21      s1.texture_crate.loadFromFile("crate_03.png");
22      s1.texture_push_block.loadFromFile("ground_04.png");
23      s1.texture_filled.loadFromFile("ground_04.png");
24      for (int i = 0; i < s1.size_y; i++) {
25          std::vector <char> v1;
26          for (int j = 0; j < s1.size_x; j++) {
27              char c;
28              cin >> c;
29              v1.push_back(c);
30              if (c == '@') {
31                  s1.track_player = {j, i};
32              } else if (c == '#') {
33                  s1.track_wall.push_back({j, i});
34              } else if (c == 'A') {
35                  s1.track_crate.push_back({j, i});
36              } else if (c == 'a') {
37                  s1.track_filled_ground.push_back({j, i});
38              }
39          }
40          s1.two_d_array.push_back(v1);
41      }
42      return cin;
43  }
44
45  void Sokoban::movePlayer(Direction player) {
46      track_player = getNextPos(player, track_player);
47      if (!(isNoWall(track_player.x, track_player.y))) {
48          track_player = getPreviousPos(player, track_player);
49      }
50      size_t size_crate = track_crate.size();
51      for (size_t i = 0; i < size_crate; i++) {
52          if (track_player == track_crate.at(i)) {
53              track_crate.at(i) = getNextPos(player, track_player);
54          }
55          // Checks to see if box is hitting a wall (Box going through wall
        fix)
56          if (!(isNoWall(track_crate.at(i).x, track_crate.at(i).y))) {
57              track_player = getNextPos(player, track_player);
58              track_crate.at(i) = getPreviousPos(player, track_player);
59              track_player = getPreviousPos(player, track_player);
60          }
61          // (Player goes through box fix)
62          if (track_player.x == track_crate.at(i).x && track_player.y ==
        track_crate.at(i).y) {
```

```
63              track_player = getPreviousPos(player, track_player);
64          }
65          // (Boxes go through each other fix)
66          sf::Vector2i first_crate_pos = track_crate.at(i);
67          for (size_t j = 0; j < size_crate; j++) {
68              if (i != j && first_crate_pos == track_crate.at(j)) {
69                  track_crate.at(i) = getPreviousPos(player, track_crate.at(i)
    );
70                  track_player = getPreviousPos(player, track_player);
71              }
72          }
73          // (Player can't go off map)
74          if (track_player.x == size_x || track_player.y == size_y ||
    track_player.x == -1
75          || track_player.y == -1) {
76              track_player = getPreviousPos(player, track_player);
77          }
78          // (Boxes can't go off map and adjusts player position)
79          for (size_t j = 0; j < size_crate; j++) {
80              if ((first_crate_pos.x == size_x || first_crate_pos.y == size_y
81              || first_crate_pos.x == -1 || first_crate_pos.y == -1) && i != j
    ) {
82                  first_crate_pos = track_crate.at(i);
83                  first_crate_pos = getPreviousPos(player, first_crate_pos);
84                  track_crate.at(i) = first_crate_pos;
85                  track_player = getPreviousPos(player, track_player);
86              }
87          }
88      }
89  }
90
91
92  sf::Vector2i Sokoban:: getNextPos(Direction player, sf::Vector2i pos) {
93      switch (player) {
94          case Up:
95              texture_player.loadFromFile("player_08.png");
96              pos.y--;
97              break;
98
99          case Down:
100             texture_player.loadFromFile("player_05.png");
101             pos.y++;
102             break;
103
104         case Right:
105             texture_player.loadFromFile("player_17.png");
106             pos.x++;
107             break;
108
109         case Left:
110             texture_player.loadFromFile("player_20.png");
111             pos.x--;
112             break;
113     }
114     return pos;
115 }
116
117 sf::Vector2i Sokoban::getPreviousPos(Direction player, sf::Vector2i pos) {
118     switch (player) {
```

```
        case Up:
            texture_player.loadFromFile("player_08.png");
            pos.y++;
            break;

        case Down:
            texture_player.loadFromFile("player_05.png");
            pos.y--;
            break;

        case Right:
            texture_player.loadFromFile("player_17.png");
            pos.x--;
            break;

        case Left:
            texture_player.loadFromFile("player_20.png");
            pos.x++;
            break;
    }
    return pos;
}
bool Sokoban::isNoWall(int x, int y) {
    for (auto w : track_wall) {
        if (w.x == x && w.y == y) {
            return false;
        }
    }
    return true;
}

bool Sokoban::isWon() const {
    size_t size = track_filled_ground.size();
    for (size_t i = 0; i < size; i++) {
        sf::Vector2i temp = track_filled_ground.at(i);
        for (size_t j = 0; j < size; j++) {
            sf::Vector2i temp2 = track_crate.at(j);
            if (temp != temp2 && i == j) {
                return false;
            }
        }
    }
    return true;
}

void Sokoban::setOriginal() {
    for (int i = 0; i < size_y; i++) {
        for (int j = 0; j < size_x; j++) {
            char c;
            c = two_d_array[i][j];
            if (c == '#') {
                track_wall.pop_back();
            } else if (c == 'A') {
                track_crate.pop_back();
            } else if (c == 'a') {
                track_filled_ground.pop_back();
            }
        }
    }
```

```
    for (int i = 0; i < size_y; i++) {
        std::vector <char> v1;
        for (int j = 0; j < size_x; j++) {
            char c;
            c = two_d_array[i][j];
            if (c == '@') {
                track_player = {j, i};
            } else if (c == '#') {
                track_wall.push_back({j, i});
            } else if (c == 'A') {
                track_crate.push_back({j, i});
            } else if (c == 'a') {
                track_filled_ground.push_back({j, i});
            }
        }
        two_d_array.push_back(v1);
    }
}
int Sokoban::height() {
    return size_y;
}
int Sokoban::width() {
    return size_x;
}
void Sokoban::draw(sf::RenderTarget &window, sf::RenderStates states) const
   {
    sf::Sprite s;
    for (int i = 0; i < size_y; i++) {
        for (int j = 0; j < size_x; j++) {
            char c = two_d_array[i][j];
            if (c == '.') {
                s.setTexture(texture_ground);
            } else if (c == '@') {
                s.setTexture(texture_ground);
            } else if (c == '#') {
                s.setTexture(texture_wall);
            } else if (c == 'A') {
                s.setTexture(texture_ground);
            } else if (c == 'a') {
                s.setTexture(texture_push_block);
            } else if (c == '1') {
                s.setTexture(texture_filled);
            }
            s.setPosition(j * GRID_SIZE, i * GRID_SIZE);
            window.draw(s);
        }
    }
    s.setTexture(texture_player);
    s.setPosition((track_player.x * GRID_SIZE), (track_player.y * GRID_SIZE)
   );
    window.draw(s);

    for (auto i : track_crate) {
        s.setTexture(texture_crate);
        s.setPosition((i.x * GRID_SIZE), (i.y * GRID_SIZE));
        window.draw(s);
    }
}
```

```
// Copyright 2023
```

```
2  // By Jake Shick
3  // main.cpp for PS4b
4  #include <vector>
5  #include <iterator>
6  #include <iostream>
7  #include <fstream>
8  #include <string>
9  #include <algorithm>
10 #include <SFML/Graphics.hpp>
11
12 const int GRID_SIZE = 64;
13 enum Direction {
14   Up,
15   Right,
16   Left,
17   Down
18 };
19
20 class Sokoban : public sf::Drawable {
21  public:
22   Sokoban();
23   Sokoban(int h, int w, std::ifstream &file);
24   friend std::istream &operator >>(std::istream &cin, Sokoban &s1);
25   void movePlayer(Direction player);
26   bool isWon() const;
27   sf::Texture getText(int i, int j);
28   sf::Vector2i getNextPos(Direction player, sf::Vector2i pos);
29   sf::Vector2i getPreviousPos(Direction player, sf::Vector2i pos);
30   bool isNoWall(int x, int y);
31   void setOriginal();
32   int height();
33   int width();
34
35  private:
36   int size_x;
37   int size_y;
38   sf::Texture texture_ground;
39   sf::Texture texture_player;
40   sf::Texture texture_wall;
41   sf::Texture texture_crate;
42   sf::Texture texture_push_block;
43   sf::Texture texture_filled;
44   sf::Vector2i track_player;
45   std::vector <sf::Vector2i> track_wall;
46   std::vector <sf::Vector2i> track_crate;
47   std::vector <sf::Vector2i> track_filled_ground;
48   void virtual draw(sf::RenderTarget &window, sf::RenderStates states) const
       ;
49   std::vector <std::vector<char> > two_d_array;
50 };
```

6 PS5: DNA Sequence

6.1 Assignment Overview

Project 5 involved comparing two DNA strings to determine the optimal distance and alignment. This process allowed us to assess the degree of dissimilarity or similarity between two binary strings. The cost signifies the distance required to reach the last character, with a sequence of 1's, 0's, and 2's representing the matched characters. For instance, if the characters T and A were matched, the cost would be 1. However, if T was followed by no subsequent character, the cost would be 2. Lastly, if the characters T and T were matched, the cost would be 0. These comparisons were conducted iteratively until the last character of both strings was reached.

6.2 Design Process

For Project 5, I applied what I learned from Projects 1, 2, and 3, and manually calculated all the necessary steps for this assignment. The approach I took involved envisioning a matrix with a zero positioned at the bottom right. This zero served as the starting point, and values increased by two for each square along the bottom horizontal and above the zero. Once this matrix was populated on the bottom horizontal row, and the vertical right row I examined each square, moving one step at a time, and selected the path with the lowest penalty—whether 0, 1, or 2. I then filled in the corresponding squares until reaching the top left, which represented the total distance traveled. With this visualization in mind, I proceeded to implement the algorithm to mimic these steps.

6.3 Key Algorithms, Data Structures, Objected-Oriented Designs, etc.

I opted to utilize a vector of vectors of integers to represent the chart. This choice was motivated by treating the matrix as a series of rows and columns to be traversed and initialized progressively until the end. Notably, I commenced the traversal from the end of the matrix (bottom right), in contrast to starting at the beginning (top left). To navigate and populate the elements, I implemented two for loops to represent the horizontal and vertical axes, multiplying the indices by two to correspond to the base horizontal and vertical values. Subsequently, I calculated the minimum of the three adjacent squares and initialized the current matrix position with this minimum value. This process continued until the entire matrix was populated.

6.4 Terminal Output

```
Edit distance = 12
a a 0
- z 2
- z 2
b b 0
c c 0
d d 0
e e 0
f f 0
g g 0
h h 0
i i 0
z - 2
z - 2
z - 2
z - 2
j j 0
k k 0
l l 0
m m 0
```

```
n n 0
o o 0
p p 0
```

6.5 What I Learned

This project provided valuable insights into string manipulation and matrix traversal. Undoubtedly, the most challenging aspect was visualizing how to access specific elements in the matrix and initialize them appropriately. On the whole, the experience significantly enhanced my understanding of two-dimensional arrays and served as a beneficial introduction to dynamic programming concepts.

6.6 Sources Used in Projects

- No Sources Used.

6.7 Challenges

In this assignment, my main challenges revolved around the alignment function and accurately calculating the time required for each sequence. The conditional checks in my loop, particularly in the while loop and if statements, were not only messy but prone to constantly checking out of bounds. As a solution, I ended up going with using an AND condition in my while loop instead of an OR, which provided a temporary fix for most alignments but didn't address all cases. Reflecting on the experience, if given the opportunity to redo this assignment, I would either restart completely or explore alternative data structures. This strategic shift could help address the issues encountered, and have a proper alignment.

6.8 Codebase

```
CC = g++
CFLAGS = --std=c++17 -Wall -Werror -pedantic -g
LIB = -lsfml-graphics -lsfml-audio -lsfml-window -lsfml-system -
   lboost_unit_test_framework
# Your .hpp files
DEPS = EDistance.hpp
# Your compiled .o files
OBJECTS = EDistance.o
# The name of your program
PROGRAM = EDistance

.PHONY: all clean lint

all: $(PROGRAM)  test $(PROGRAM).a #test

%.o: %.cpp $(DEPS)
    $(CC) $(CFLAGS) -c $<

%.o: %.cpp $(DEPS)
    $(CC) $(CFLAGS) -c $<

$(PROGRAM): main.o $(OBJECTS)
    $(CC) $(CFLAGS) -o $@ $^ $(LIB)

test:  test.o $(OBJECTS)
    $(CC) $(CFLAGS) -o $@ $^ $(LIB)

$(PROGRAM).a: $(OBJECTS)
    ar rcs $@ $^

```

```
clean:
    rm *.o $(PROGRAM) $(PROGRAM).a

lint:
    cpplint *.cpp *.hpp
```

```
// <Copyright 2023> Chisom Ukaegbu and Jake Shick
#include <iostream>
#include <fstream>
#include <string>
#include <SFML/System.hpp>
#include "EDistance.hpp"

int main(int argc, char* argv[]) {
    std::string seq1, seq2;
    std::cin >> seq1;
    std::cin >> seq2;
    int Distance;
    EDistance ed(seq1, seq2);
    Distance = ed.optDistance();
    std::string align = ed.alignment();
    std::cout<< "Edit distance = " << Distance << std::endl;
    std::cout << align;
    return 0;
}
```

```
#include <iostream>
#include <string>
#include <vector>
#include <sstream>
#include "EDistance.hpp"

EDistance::EDistance(std::string _seq1, std::string _seq2) {
    seq1 = _seq1;
    columns = _seq1.size();
    seq2 = _seq2;
    rows = _seq2.size();
    std::flush(std::cout);
    matrix.resize(columns + 1, std::vector<int>(rows + 1, 0));
}
int EDistance::penalty(char a, char b) {
    return a != b ? 1 : 0;
}

int EDistance::optDistance() {
    for (int i = 0; i < columns; i++) {
        matrix[i][rows] = two * (columns - i);
    }
    for (int i = 0; i < rows; i++) {
        matrix[columns][i] = two * (rows - i);
    }
    for (int i = columns - 1; i >= 0; i--) {
        for (int j = rows - 1; j >= 0; j--) {
            matrix[i][j] = min3(matrix[i + 1][j + 1] + penalty(seq1[i], seq2
   [j]),
            matrix[i + 1][j] + 2,
            matrix[i][j + 1] + 2);
        }
    }
    return matrix[0][0];
```

```
}
std::string EDistance::alignment() {
    std::string result;
    int i = 0, j = 0;
    int _rows = rows;
    int _columns = columns;
    int count = 0;
    while (i < _columns && j < _rows) {
        if (seq1[i] == seq2[j]
            && matrix.at(i).at(j) == matrix.at(i + 1).at(j + 1)) {
            result += std::string(1, seq1[i]) + " " + std::string(1, seq2[j
   ]) + " 0";
            result += "\n";
            i++;
            j++;
        } else if (seq1[i] != seq2[j] &&
            matrix.at(i).at(j) == matrix.at(i + 1).at(j + 1) + 1) {
            result += std::string(1, seq1[i]) + " " + std::string(1, seq2[j
   ]) + " 1";
            count++;
            result += "\n";
            i++;
            j++;
        } else if (matrix.at(i).at(j) == matrix.at(i + 1).at(j) + 2) {
            result += std::string(1, seq1[i]) + " - 2";
            result += "\n";
            count += 2;
            i++;
        } else if (matrix.at(i).at(j) == matrix.at(i).at(j + 1) + 2) {
            result += "- ";
            result += std::string(1, seq2[j]);
            result += " 2";
            result += "\n";
            count += 2;
            j++;
        }
    }
    return result;
}

int EDistance::min3(int a, int b, int c) {
    int temp = 0;
    if (a > b)
        temp = b;
    else
        temp = a;

    if (temp > c)
        return c;

    else
        return temp;
}
```

```
// Copyright 2023
// main.cpp for PS5 Jake Shick and Chisom Ukaegbu
#pragma once
#include <iostream>
#include <string>
#include <vector>
```

```
#include <sstream>
const int two = 2;
class EDistance {
 public:
    EDistance(std::string seq1, std::string seq2);
    static int penalty(char a, char b);
    static int min3(int a, int b, int c);
    int optDistance();
    std::string alignment();
 private:
    std::string seq1;
    std::string seq2;
    int rows;
    int columns;
    std::vector<std::vector<int>> matrix;
};
```

```
// Copyright 2023 Chisom Ukaegbu and Jake Shick[legal/copyright]
#define BOOST_TEST_DYN_LINK
#define BOOST_TEST_MODULE Main
#include <boost/test/unit_test.hpp>
#include "EDistance.hpp"

BOOST_AUTO_TEST_CASE(testDistance) {
    EDistance ed("GAACC", "CAAGAC");
    int testDist = ed.optDistance();
    int correctDist = 4;
    BOOST_CHECK_EQUAL(testDist, correctDist);
}

BOOST_AUTO_TEST_CASE(testMin) {
    EDistance edMin("GAACC", "CAAGAC");
    int minimum = edMin.min3(30, 5, 9);
    BOOST_CHECK_EQUAL(minimum, 5);
}

BOOST_AUTO_TEST_CASE(testPen) {
    EDistance edPen("GAACC", "CAAGAC");
    int pen0 = edPen.penalty('a', 'b');
    int pen1 = edPen.penalty('a', 'a');
    BOOST_CHECK_EQUAL(pen0, 1);
    BOOST_CHECK_EQUAL(pen1, 0);
}

BOOST_AUTO_TEST_CASE(testAlign) {
    EDistance edAlign("AATA", "TA");
    edAlign.optDistance();
    std::string tAlign = edAlign.alignment();
    std::string correctAlign1 = "A - 2\nA - 2\nT T 0\nA A 0\n";
    BOOST_CHECK_EQUAL(tAlign, correctAlign1);
}
```

7 PS6: RandWriter

7.1 Assignment Overview

For Project 6, the task involved creating a Markov model of order k to analyze the frequency of specific words or characters following others. The assignment required generating phrases or sentences based on the value of k, aiming to produce output that is either similar or exact to the original strings provided. In this case, the input data was taken from Tom Sawyer, and the goal was to determine the patterns of word and character sequences in the text.

7.2 Design Process

In the initial phase, I addressed the task of obtaining the frequency of k-grams, or sequences of letters or characters. This involved checking whether and when these k-grams appeared in the text, and returning the count to the caller. I then concentrated on implementing the kRand function, which was responsible for generating a random character based on the probabilities of it following a given string or character. However, the generate function posed the greatest challenge. It led to an infinite loop and errors when attempting to iterate through the dictionary.

7.3 Key Algorithms, Data Structures, Objected-Oriented Designs, etc.

I used a multidimensional unordered map to model the dictionary. The primary key is a string, signifying the line or text retrieved from the file. Within this structure, the inner unordered map contains a character as keys, representing potential characters following the original string or sequence. Each character is associated with an integer indicating the frequency of its occurrence.

7.4 Terminal Output

```
Value of K = 2
The frequency in which the letter e appeared = 1
The most likely letter to follow the string "th" = e
```

7.5 What I Learned

This project significantly improved my proficiency with dictionaries, enabling me to confidently traverse and manipulate them. I now feel more skilled at searching through large text datasets to locate specific information, addressing a challenge I faced in earlier computing classes.

7.6 Sources Used in Projects

- Map Help: `https://stackoverflow.com/questions/1939953/how-to-find-if-a-given-key-ex`
- Spacing Issues: `https://stackoverflow.com/questions/5838711/stdcin-input-with-spaces`

7.7 Challenges

I encountered challenges primarily with the generate function in this assignment, facing issues such as never-ending loops and generating text that didn't make much sense. If given the opportunity to redo the project, I would have opt to start the generate function from scratch to identify the areas where it was causing the most issues.

7.8 Codebase

```
CC = g++
CFLAGS = --std=c++17 -Wall -Werror -pedantic -g
LIB = -lsfml-graphics -lsfml-audio -lsfml-window -lsfml-system -
   lboost_unit_test_framework
# Your .hpp files
DEPS = RandWriter.hpp
# Your compiled .o files
OBJECTS = RandWriter.o
# The name of your program
PROGRAM = TextWriter

.PHONY: all clean lint

all: $(PROGRAM) test $(PROGRAM).a

%.o: %.cpp $(DEPS)
    $(CC) $(CFLAGS) -c $<

%.o: %.cpp $(DEPS)
    $(CC) $(CFLAGS) -c $<

$(PROGRAM): TextWriter.o $(OBJECTS)
    $(CC) $(CFLAGS) -o $@ $^ $(LIB)

test:  test.o $(OBJECTS)
    $(CC) $(CFLAGS) -o $@ $^ $(LIB)

$(PROGRAM).a: $(OBJECTS)
    ar rcs $@ $^

clean:
    rm *.o $(PROGRAM) $(PROGRAM).a

lint:
    cpplint *.cpp *.hpp
```

```
// Copyright 2023
// By Jake Shick
// TextWriter.cpp for PS6
#include <iostream>
#include <string>
#include <fstream>
#include "RandWriter.hpp"

int main(int argc, char** argv) {
    int k = std::stoi(argv[1]);
    int L = std::stoi(argv[2]);
    std::string F;
    while (std::getline(std::cin, F)) {}
    RandWriter R(F, k);
    std::cout << R.generate("the", L) << std::endl;
}
```

```
// Copyright 2023
// By Jake Shick
// RandWriter.cpp for PS6
#include "RandWriter.hpp"

```

```
6  // Create a Markov model of order k from given text
7  // Assume that text has length at least k.
8  RandWriter::RandWriter(std::string text, int _k) {
9      k = _k;
10     int size = text.size();
11     text.append(text.substr(0, _k));
12     unsigned int seed = std::chrono::system_clock::now().time_since_epoch().
       count();
13     random_number.seed(seed);
14     for (int i = 0; i < size; i++) {
15         dictionary[text.substr(i, _k)][text.at(i + _k)]++;
16     }
17 }
18
19 // Order k of Markov model
20 int RandWriter::orderK() const {
21     return k;
22 }
23
24 // Number of occurences of kgram in text
25 // Throw an exception if kgram is not length k
26 int RandWriter::freq(std::string kgram) const {
27     int count = 0;
28     if (kgram.length() != (unsigned) k) {
29         throw std::out_of_range("ERROR: The size of kgram is not equal to k"
       );
30         return -1;
31     }
32     if (dictionary.find(kgram) == dictionary.end()) {
33         return 0;
34     }
35     for (auto itr : dictionary.at(kgram)) {
36         count += itr.second;
37     }
38     return count;
39 }
40
41 // Number of times that character c follows kgram
42 // if order=0, return num of times that char c appears
43 // (throw an exception if kgram is not of length k)
44 int RandWriter::freq(std::string kgram, char c) const {
45     if (kgram.length() != (unsigned) k)
46         throw std::out_of_range("ERROR: The size of kgram is not equal to k"
       );
47     if (dictionary.find(kgram) == dictionary.end())
48         return 0;
49     if (dictionary.at(kgram).find(c) == dictionary.at(kgram).end())
50         return 0;
51     return dictionary.at(kgram).at(c);
52 }
53 // dictionary.at(kgram).find(c) == dictionary.at(kgram).end()
54 // Random character following given kgram
55 // (throw an exception if kgram is not of length k)
56 // (throw an exception if no such kgram)
57 char RandWriter::kRand(std::string kgram) {
58     if (kgram.length() != (unsigned) k) {
59         throw std::out_of_range("ERROR: The size of kgram is not equal to k"
       );
60     }
```

```
    if (dictionary.find(kgram) == dictionary.end()) {
        return 0;
    }
    std::uniform_int_distribution <int> dis(0, freq(kgram));
    int count = dis(random_number);
    for (auto pair : dictionary.at(kgram)) {
        count -= pair.second;
        if (count <= 0)
            return pair.first;
    }
    return '-';
}

// Generate a string of length L characters by simulating a trajectory
// through the corresponding Markov chain. The first k characters of
// the newly generated string should be the argument kgram.
// Throw an excpetion if kgram is not of length k.
// Assume that L is at least k
std::string RandWriter::generate(std::string kgram, int L) {
    if (kgram.length() != unsigned(k)) {
        throw std::out_of_range("ERROR: The size of kgram is not equal to k"
   );
    }
    std::string str;
    char rand_char;
    str += kgram;
    for (int i = 0; unsigned(i) < (L - unsigned(k)); i++) {
        rand_char = kRand(kgram);
        str += rand_char;
    }
    return str;
}

std::ostream &operator <<(std::ostream &out, RandWriter &R) {
    out << R.k << std::endl;
    for (auto pair1 : R.dictionary) {
            out << pair1.first;
        for (auto pair2 : pair1.second) {
            out << pair2.first;
        }
    }
    return out;
}
```

```
// Copyright 2023
// By Jake Shick
// RandWriter.hpp for PS6
#pragma once
#include <iostream>
#include <unordered_map>
#include <algorithm>
#include <chrono>
#include <iterator>
#include <random>
#include <numeric>
#include <string>
class RandWriter {
 public:
    // Create a Markov model of order k from given text
    // Assume that text has length at least k.
```

```
    RandWriter(std::string text, int k);

    // Order k of Markov model
    int orderK() const;

    // Number of occurences of kgram in text
    // Throw an exception if kgram is not length k
    int freq(std::string kgram) const;

    // Number of times that character c follows kgram
    // if order=0, return num of times that char c appears
    // (throw an exception if kgram is not of length k)
    int freq(std::string kgram, char c) const;

    // Random character following given kgram
    // (throw an exception if kgram is not of length k)
    // (throw an exception if no such kgram)
    char kRand(std::string kgram);

    // Generate a string of length L characters by simulating a trajectory
    // through the corresponding Markov chain. The first k characters of
    // the newly generated string should be the argument kgram.
    // Throw an excpetion if kgram is not of length k.
    // Assume that L is at least k
    std::string generate(std::string kgram, int L);

    friend std::ostream &operator <<(std::ostream &out, RandWriter &R);

 private:
    std::unordered_map <std::string, std::unordered_map <char, int> >
   dictionary;
    std::minstd_rand0 random_number;
    int k;
};
```

```
// Copyright 2022
// By Dr. Rykalova
// Editted by Dr. Daly
// Also editted by Jake Shick
// test.cpp for PS6
// updated 5/12/2022

#include <iostream>
#include <string>
#include "RandWriter.hpp"
#define BOOST_TEST_DYN_LINK
#define BOOST_TEST_MODULE Main
#include <boost/test/unit_test.hpp>

BOOST_AUTO_TEST_CASE(testkOrder) {
    int k = 2;
    std::string str = "It was the best of times";
    RandWriter R(str, k);
    BOOST_CHECK_EQUAL(k, R.orderK());
}

// Wanted to make sure would get g to see if I had string of all g's
BOOST_AUTO_TEST_CASE(testkRand) {
    int k = 1;
    std::string str = "ggg";
```

```
26      RandWriter R(str, k);
27      char a = R.kRand("g");
28      BOOST_CHECK_EQUAL(a, 'g');
29  }
30
31  BOOST_AUTO_TEST_CASE(testFreq1) {
32      int k = 1;
33      std::string str = "gagggagaggcgagaaag";
34      RandWriter R(str, k);
35      int i = R.freq("g");
36      BOOST_CHECK_EQUAL(i, 10);
37  }
38
39  BOOST_AUTO_TEST_CASE(testFreq2) {
40      int k = 1;
41      std::string str = "gagggagaggcgagaaag";
42      RandWriter R("ga", k);
43      BOOST_REQUIRE_THROW(R.freq(str), std::out_of_range);
44  }
45
46  BOOST_AUTO_TEST_CASE(testGen) {
47      int k = 1;
48      int L = 11;
49      std::string str = "gagggagaggcgagaaag";
50      RandWriter R(str, k);
51      std::string str2 = R.generate("g", L);
52      BOOST_CHECK_EQUAL(11, str2.size());
53  }
```

8 PS7: Kronos Log Parsing

8.1 Assignment Overview

Project 7 was the final assignment of the semester which had us analyze a log file from a kronos clock and check to see which boot ups were successful, and which ones were not. We would then output the intial startup time, the line number, and which device it booted up. The assignment was used as an introduction to regex, an API used for searching through large documents to get specific information from them.

8.2 Design Process

The first thing I did for this was figure out what my regexes needed to look like in order to find the specific line. For the boot startup I created a regex that checked to see a line that had four digits displayed first followed by two more digits, followed by two more digits. I then repeated that since it has that on the same line twice, and included the specific information used for that line. This way when the entire 400,000 plus lines of logs was being read through a while loop, I could pin point which ones I actually needed to find line by line.

8.3 Key Algorithms, Data Structures, Objected-Oriented Designs, etc.

For this, I ended up using the built in regex functions, regex search and regex match to check to see if any of my regexes matched the strings. The regex search function was used for getting the date of when the boot was either completed or started. Knowing this, we could then check to see the time it took from startup to its completion. The regex match function was just used to actually check to see if we found a string that matched our regex, and if so we would go into the loop and complete some action.

8.4 Kronos Output

```
=== Device boot ===
435369(device1_intouch.log): 2014-03-25 19:11:59 Boot Start
435759(device1_intouch.log): 2014-03-25 19:15:02 Boot Completed
    Boot time: 183000ms

=== Device boot ===
436500(device1_intouch.log): 2014-03-25 19:29:59 Boot Start
436859(device1_intouch.log): 2014-03-25 19:32:44 Boot Completed
    Boot time: 165000ms

=== Device boot ===
440719(device1_intouch.log): 2014-03-25 22:01:46 Boot Start
440791(device1_intouch.log): 2014-03-25 22:04:27 Boot Completed
    Boot time: 161000ms

=== Device boot ===
440866(device1_intouch.log): 2014-03-26 12:47:42 Boot Start
441216(device1_intouch.log): 2014-03-26 12:50:29 Boot Completed
    Boot time: 167000ms

=== Device boot ===
442094(device1_intouch.log): 2014-03-26 20:41:34 Boot Start
442432(device1_intouch.log): 2014-03-26 20:44:13 Boot Completed
    Boot time: 159000ms

=== Device boot ===
443073(device1_intouch.log): 2014-03-27 14:09:01 Boot Start
443411(device1_intouch.log): 2014-03-27 14:11:42 Boot Completed
    Boot time: 161000ms
```

8.5 What I Learned

This project helped introduce me into regex and a very interesting way for me. Like I had previously stated for one of my other assignments, going through large files to search for specific information has always been a challenge for me, but with this new tool, I feel I won't have many issues with that for future assignments and projects.

8.6 Sources Used in Projects

- Regular Expressions Help:
  `https://www.geeksforgeeks.org/smatch-regex-regular-expressions-in-c/`

8.7 Codebase

```
CC = g++
CFLAGS = --std=c++17 -Wall -Werror -pedantic -g
LIB = -lsfml-graphics -lsfml-audio -lsfml-window -lsfml-system -
   lboost_unit_test_framework -lboost_date_time -lboost_regex
# Your .hpp files
DEPS =
# Your compiled .o files
OBJECTS = main.o
# The name of your program
PROGRAM = ps7

.PHONY: all clean lint

all: $(PROGRAM) $(PROGRAM).a

%.o: %.cpp $(DEPS)
    $(CC) $(CFLAGS) -c $<

$(PROGRAM): main.o $(OBJECTS)
    $(CC) $(CFLAGS) -o $@ $^ $(LIB)

$(PROGRAM).a: $(OBJECTS)
    ar rcs $@ $^

clean:
    rm *.o $(PROGRAM) $(PROGRAM).a

lint:
    cpplint *.cpp *.hpp
```

```
// Copyright 2023
// By Jake Shick
// Regex for PS7
#include <iostream>
#include <string>
#include <fstream>
#include <sstream>
#include <boost/regex.hpp>
#include <boost/date_time/posix_time/posix_time.hpp>

int main(int argc, char** argv) {
    std::ifstream inData;
    std::ofstream outData;
```

```
14      boost::regex log("^\\d{4}-\\d{2}-\\d{2} \\d{2}:\\d{2}:\\d{2}: "
15      "\\(log\\.c\\.166\\) server started $");
16      boost::regex oejs("^(\\d{4}-\\d{2}-\\d{2} \\d{2}:\\d{2}:\\d{2}\\.\\d+)"
17      ":INFO:oejs\\.AbstractConnector:Started SelectChannelConnector@"
18      "(\\d+\\.\\d+\\.\\d+\\.\\d+):(\\d+)$");
19      boost::regex date_of_complete("^(\\d{4}-\\d{2}-\\d{2} "
20      "\\d{2}:\\d{2}:\\d{2})");
21      boost::posix_time::ptime start_time;
22      std::string str;
23      int count = 1;
24      bool flag = false;
25      inData.open(argv[1]);
26      if (!inData) {
27          std::cout << "NOPE" << std::endl;
28          return -1;
29      }
30      std::string name_of_log_file = argv[1];
31      name_of_log_file.append(".rpt");
32      outData.open(name_of_log_file);
33      if (!outData) {
34          std::cout << "NOPE" << std::endl;
35          return -1;
36      }
37      while (getline(inData, str)) {
38          if (regex_match(str, log)) {
39              boost::smatch m;
40              if (flag) {
41                  outData << "**** Incomplete boot ****" << std::endl << std::
    endl;
42              } else {
43                  flag = true;
44              }
45              regex_search(str, m, date_of_complete);
46              outData << "=== Device boot ===" << std::endl;
47              outData << count << "(" << argv[1] << "): " << m.str() <<
48                  " " << "Boot Start" << std::endl;
49              std::stringstream ss(m.str());
50              boost::posix_time::time_facet* ofacet =
51                  new boost::posix_time::time_facet("%Y-%m-%d %H-%M-%S");
52              boost::posix_time::time_input_facet* ifacet =
53                  new boost::posix_time::time_input_facet("%Y-%m-%d %H-%M-%S")
    ;
54              ss.imbue(std::locale(ss.getloc(), ifacet));
55              std::cout.imbue(std::locale(std::cout.getloc(), ofacet));
56              ss >> start_time;
57          }
58          if (regex_match(str, oejs)) {
59              boost::posix_time::ptime end_time;
60              boost::smatch m;
61              regex_search(str, m, date_of_complete);
62              outData << count << "(" << argv[1] << "): "
63              << m.str() << " " << "Boot Completed" << std::endl;
64              std::stringstream ss(m.str());
65              boost::posix_time::time_facet* ofacet =
66                  new boost::posix_time::time_facet("%Y-%m-%d %H-%M-%S");
67              boost::posix_time::time_input_facet* ifacet =
68                  new boost::posix_time::time_input_facet("%Y-%m-%d %H-%M-%S")
    ;
69              ss.imbue(std::locale(ss.getloc(), ifacet));
```

```
70              std::cout.imbue(std::locale(std::cout.getloc(), ofacet));
71              ss >> end_time;
72              boost::posix_time::time_duration diff = end_time - start_time;
73              outData << "    " << "Boot time: "
74              << diff.total_milliseconds() << "ms" << std::endl << std::endl;
75              flag = false;
76          }
77          count++;
78      }
79      inData.close();
80      outData.close();
81      return 0;
82  }
```